

A Survey of Data Acquisition & Analysis Software Tools *(including Visual Programming.)*

Ed Baroth, Ph. D., Manager, and
Chris Hartsough, Lee Johnson, Jim McGregor, Mag Powell-Mccks, Amy Walsh,
George Wells, Seth Chazanoff and Ted Brunzie, Members of Technical Staff,
Measurement Technology Center
Jet Propulsion Laboratory, California Institute of Technology

PART 1

INTRODUCTION

Part of the Jet Propulsion Laboratory's (JPL's) instrumentation Section, the Measurement Technology Center (MTC) evaluates data acquisition hardware and software products for inclusion into the instrument Loan Pool, which are then made available to JPL experimenters. As such, it acts as a focus for off-the-shelf products. The MTC also configures turn-key measurement systems that include integrated sensors, signal conditioning, data simulation, acquisition, analysis, display and control capabilities.^{1,2}

The purpose of this article is to discuss several different types of software tools the MTC has used to develop such systems. To help demonstrate the capabilities of each package, a sample test program was developed and will be described. Figure 1 shows five categories of toolsets on a 'crossroads' sign to introduce the reader to the confusing array of programming tools available. This article should help reduce the confusion.

The signs express separate approaches that can be taken to performing data acquisition, analysis and display: the 'traditional' approach of text-based BASIC or C programming, the combination of graphical user interface (GUI) tools with text-based programming approach (with or without instrument support), or the visual programming approach.

Code is written in visual programming languages by creating and connecting icons. These icons represent functions (subroutines) and are connected by 'wires' that are paths which variables travel from one function to the next. Visual 'code' is actually the diagram of icons and wires rather than a text file of sequential instructions. Although available since 1986, visual programming software is just now becoming popular due to maturity, use in general purpose applications and availability across platforms.^{3,4,5}

TESTED SOFTWARE PACKAGES

This article evaluates some of the general purpose data acquisition and analysis packages: LabVIEW and LabWindows from National Instruments, Visual Engineering Environment (VEE) from Hewlett-Packard and WaveTest VIP from Wavetek. Table 1 shows the version, platforms and configurations. Also discussed will be NI Power from Signal Technology inc., LabTech Notebook from Laboratory Technologies and ATEasy from Geotest. There are certainly more packages available, with the list growing daily, but not all fit the needs of the MTC.

The MTC acts as a service to JPL engineers and scientists, who have widely diverse needs, from simple one day programming tasks to projects that require many months and several engineers. In selecting the tools to use for providing this service and this article, tools are examined with the following questions in mind:

General purpose system approach

The software must be as general purpose as possible. It must command a variety of interface cards to acquire, analyze, display data and control different kinds and types of instrumentation. This includes the ability to easily and quickly create instrument drivers when not available. It is also important that a graphical user interface (GUI) can be developed. The computer acts as the 'window' to the experiment and information from all the instruments should be integrated on the screen. The software should act in concert with the hardware to form a 'system solution.' Software packages that

fit one niche, however well, (e.g., convert a PC into an oscilloscope, or do DSP or process control), but cannot be expanded into a general purpose system, are of no use to the MTC.

Quickly learned, easy to use

In judging software tools, the MTC is interested in the time to learn and the ease of use of software tools. Of course, the more a tool is used, the easier it is to use, making these criteria somewhat subjective. What isn't subjective is whether the software requires an experienced BASIC or C programmer, uses visual programming, encourages or forces object-oriented software development.

Available on more than one platform

It is highly advantageous to use a tool that is available on more than one platform, especially if the applications themselves are transportable. Applications developed for one customer's Mac, can be modified to run on a PC, eliminating duplicity of effort. It also reduces learning time, as one tool need be learned, instead of different tools for different platforms.

Generates a stand alone, run-time version

The program is usually developed on a computer in the MTC laboratory before delivery to the customer. It's desirable that delivered turn-key systems not require a copy of the developmental tool used to generate the program. The generated program should be a stand-alone program, or at least require only a cheaper run-time package for execution.

Well established with a good track record

From previous experience, the MTC is wary of new programs and newly released versions of established programs. Delays resulting from undiscovered 'features' in a development tool can interfere with the timely delivery of a system. Good support from an established manufacturer is necessary, as is training because the MTC frequently delivers systems to users who make their own day-to-day changes.

Hardware or software keys not required

After losing weeks of work due to lost, broken, or non-working keys, the MTC no longer supports development tools that require either a hardware or software key (not simply a license).

Platform	Minimum Configuration Recommended: ()	Configuration Used
IBM PC/Clone		
LabWindows Version 2.3a \$1995	IBM PC AT or compatible 80286, math coprocessor, 2 Mb memory, 8 Mb hard disk space, EGA, VGA, Super VGA, or Hercules graphics adapter, mouse, DOS 3.1	80486/33, 16 Mb memory, 630 Mb hard disk, VGA, mouse, DOS 5.0
LabVIEW for Windows Version 2.5.2 \$1995	IBMPC AT or compatible 80386/25, math coprocessor, 8 Mb memory, 16 Mb hard disk space, EGA, VGA, Super VGA, or Hercules graphics adapter, mouse, DOS 3.1, Windows 3.1	80486/33, 16 Mb memory, 630 Mb hard disk, VGA, mouse, DOS 5.0, Windows 3.1
WaveTest VIP Version 1.0 \$695	IBMPC AT or compatible 80286/25, (80386), math coprocessor, 3 Mb memory, 5 Mb hard disk space, EGA, (VGA), mouse, DOS 3.1, Windows 3.1	80386/25, 8 Mb memory, 220 Mb hard disk, VGA, mouse, DOS 5.0, Windows 3.1, Visual BASIC
VEE Windows Version 2.0 \$1995 RunTime: \$495	IBM PC or compatible 80386 /33DX, math coprocessor, (80486DX), 8 Mb memory, (16 Mb), 15 Mb hard disk space, VGA 640x480 16-color, (SVGA 1024x768 256 color), mouse, DOS 5.0, Windows 3.1	80486/33, 16 Mb memory, 630 Mb hard disk, VGA, mouse, DOS 5.0, Windows 3.1
Macintosh		
LabVIEW Version 2.2.1 \$1995	Any Macintosh computer with at least 4MB of RAM, hard disk, System 6.0.3 or later, 5 Mb RAM for System 7.0 or later, 20 Mb hard disk space. 68040 processor required for Version 3.0	Quadra 950, 24 Mb RAM, 400 Mb hard disk, System 7.0.1, and Mac IIx, 20 Mb RAM, 160 Mb hard disk, System 7.0.1

“ Hewlett Packard HP VEE Version 2.0 \$6000	HP 9000 Series 300, 400, 700, 12 Mb RAM, (16 Mb), HP-UX Model 382, 32 Mb RAM, 420 8.x, X Windows 11.4, 5 Mb Swap, 20 Mb hard disk space SCSI hard disk
sun Microsystems LabVIEW (Spare) Version 2.5.2 \$4000	Sparc1 or later, supports OpenWindows 3 or XI 1R4 or 1-U, Spare 2GX, 32 Mb RAM, 880 Mb 24 Mb RAM, 32 Mb disk swap, 12 Mb hard disk space hard disk
VEE (Spare) Version 2.0 \$6000	Sparc1 or later, Sun OS 4.1.2 or 4.1.3, XI 1 or Open Systems Spare 2GX, 32 Mb RAM, 880 Mb 2.0 or 3.0, 12 Mb RAM (24 Mb), 5 Mb Swap, 20 Mb hard disk hard drive space

TABLE 1. Tested Software Packages

GENERAL NOTES ON DRIVERS

An important part of any data acquisition/instrumentation program is the ‘driver’ section. Without drivers, which have come to mean software that enables communication between the computer and the instrument, there would be no data to acquire, analyze, or display. But there are different kinds of ‘drivers’: drivers integrated into the operating system, interface drivers, drivers for special purpose internal I/O boards (such as analog to digital and digital to analog converters), and higher level instrument drivers.

The interface driver is used for controlling bus interfaces, e.g., IEEE-488, RS-232, VXI or VME. These low-level protocols are the usual means of communicating with instruments or other devices. Drivers contain functions for controlling communications through the interface. There is also some method provided for checking interface errors. This type of driver is used as a tool for writing the higher level instrument drivers. Interface drivers usually come with the GPIB or 1<S-232 card, or they may be included in the user’s software. It is seldom that this type of driver has to be created by the user, and when this does occur, it is not a task for the uninitiated. All the programs tested contain this type of 10 W-1CVCI driver.

Special purpose I/O boards nearly always come with a set of low-level protocols. Since these are the most difficult to write, the prudent user should very carefully question the manufacturer of the board, especially as to its compatibility with the application package being used. It is not unusual for the cost of developing this type of ‘driver’ to be many times the cost of the board. All the packages tested contain drivers for a wide range of boards, although some of them only support boards from the same manufacturer.

What is generally called an instrument driver is really no more than a collection of instrument control functions passed to an interface driver. The commands perform the following tasks: verify the interface to the instrument, initialize or calibrate the instrument, configure it for the desired operation, start it performing the operation, check its status, and read its data. The instrument driver may be included with the instrument or it may be available separately. In either case, however, an instrument driver written for one development tool will not generally work with another. (C drivers won’t work with BASIC, LabVIEW drivers won’t work with WaveTest VIP, etc.)

If the driver for the instrument is not available, it must be created. Software packages that present a good environment for creating drivers are important because driver development can be a time consuming and difficult task if not adequately supported by the software tools. A friendly development environment can make writing the instrument driver almost trivial, assuming the instrument manual contains all the relevant and correct information (not always a good assumption).

National Instruments, Hewlett Packard and Wavetek provide libraries of instrument drivers for their respective software. National’s and Wavetek’s lists are more extensive and general, containing instruments from many manufacturers, while Hewlett Packard’s list is heavy on, well, Hewlett Packard instruments. Murphy’s Law applies, however -- no matter how extensive the library, your particular instrument will not be included. That’s the bad news. The good news is that with any of the tested packages, writing a driver is really no big deal (except maybe for the first one).

In most cases, the user only needs half a dozen or so functions of an instrument’s capability. Writing a driver is simply tapping those functions, not every function the instrument can perform. Once written, the driver becomes integrated into the program. Vast libraries of instrument drivers are of great importance only to the beginner. Advanced users usually copy parts of the library driver into

their own program, or most often write their own from scratch. This way, only the commands used are included, and the complexities of large drivers are avoided.

PROGRAMMING WITH THE BASIC LANGUAGE

With the arrival of the Hewlett Packard Interface Bus (HPIB) and HP BASIC (also known as Rocky Mountain BASIC), the average engineer could develop an application in a fraction of the time it took using PL/M or assembly languages. All the low level HPIB driver calls were built into HP BASIC. Of course HP BASIC was only available for HP's computers and only HP instruments had an HPIB bus. However, this method of communicating between a computer and an instrument proved to be so popular that other manufacturers began to add HPIB bus control to their own instruments, and non-HP computer interface cards began to show up for other computers. This instrumentation bus became so popular that it was adopted as the industry standard IEEE-488 General Purpose Interface Bus (GPIB) and now instruments using it are commonplace.

Programs for instrument control were originally written in some BASIC version on an IBM PC or clone by the engineer that was to use them. They were menu driven with no graphical interface capability and were typically rather unsophisticated. The programs did nothing more than initialize the instrument with a set of fixed parameters, read values from the instrument at predefined intervals, and write the data to a file. The data was then processed and plotted using another program, usually a spreadsheet program. Only when the engineer could afford a programmer for a few weeks did the programs contain anything that resembled a graphical user interface. Even then, the user interface was, at best, little more than data being plotted in real-time as it was acquired. Programming in one of the menu-graphical BASIC languages is still the same.

OBJECT-ORIENTED PROGRAMMING

The next breakthrough in developing data acquisition systems came with the introduction of object-oriented programming methodology. Object-oriented design is a bottom-up method of structured programming, where the analyst begins with the fine details and works up toward the main level of the program. It models a program as a set of cooperating objects that includes both data and functions. Because it models both behavioral and information complexity, the program is much better organized than if it were simply well structured. This allows programs to be easier to understand, debug, maintain and evolve. Object-oriented design lends itself to team programming and code reuse.

The most common object-oriented programming tool is C++.

VISUAL BASIC

One of the BASIC programs that has become popular is Microsoft's Visual BASIC. There are two versions of Visual BASIC -- one that requires Microsoft's Windows and one that is a stand-alone development tool for DOS. The use of the Visual BASIC for DOS Professional Edition will be described here. The DOS version was chosen over the Windows version because it can generate stand-alone, compiled code that requires neither Windows nor Visual BASIC for execution. This means that the application program can be executed on another machine with a minimal amount of memory and without the extra cost of other run-time supporting software. An important side effect is that Visual Basic for DOS applications runs faster than Visual BASIC for Windows applications (actually, they don't run slower in Windows, the delays are due to the overhead of the Windows environment).

The first step in development of an application is the design and creation of the visual user interface, in other words, what the user will see on the screen. This consists of a panel of objects such as switches and push buttons, textual display boxes, selection list boxes, and plots or graphs. These and other objects (called controls) are placed inside windows that the users see. The programmer draws the controls on a blank window that will become the user interface. After the interface has been created, the objects on it will automatically recognize user actions such as mouse movements and button clicks.

Nothing similar to conventional programming occurs until after the GUI has been created. Even then, the programming takes place in small unconnected segments. Each control in the window has a function associated with it. This function, determined by the programmer, is coded using BASIC in the traditional way. The code, however, will only be executed when the control is activated with the movement of the mouse or with the click of a button. The selection of a control is called an event.

Conventional programs run from the top, down. The only change in the flow of the program is determined by the original programmer. Visual BASIC, on the other hand, works in a completely different way. The Visual Basic program consists of a set of independent segments of code that are activated by the user. The choice of functions and the order in which they are executed are determined by the user.

If you have used the Windows version of Visual Basic, you will be disappointed in the DOS version. The panels do not make use of the visual mode of the computer's interface card. Instead, the controls are constructed from the extended characters of DOS's ANSI set. This causes the control's size to be an integral multiple of character lines in height and an integral multiple of characters in width. This is the price paid for keeping VBDOS small.

Visual BASIC is best used as a tool to develop GUI's for a data acquisition package that can take advantage of it.

SAMPLE PROGRAM REQUIREMENTS

A sample data acquisition and analysis program was written using each package. The program requirements were chosen to demonstrate the following activities: Developing 'drivers' that allow communication with instruments through GPIB, commanding and controlling instruments, acquiring, analyzing and plotting data from instruments, and developing a software user interface integrated with the hardware.

The sample program should give the reader a feel for problems encountered during application development. Also, the screen design tools and visual presentation features of each tool are demonstrated.

The two instruments chosen were a signal generator and an analog to digital converter. The application program controlled the wave type, amplitude, and frequency of the signal generator and the sample rate, number of samples, and triggering of the A/D converter. The actual instruments and their programmed operations were:

Wavetek Model 23 Function Generator:
Wave type: Sine, square, or triangle.
Amplitude: Min. = .01 volt, Max. = 10 volt.
Frequency: Min. = 10 Hz, Max. = 999 Hz.

IOtech ADC488/16A Analog to Digital converter:
Sampling rate: Min. = 1 kHz, Max. = 10 kHz
Number of samples upon triggering = 1024

The sample program, upon command, acquired 1024 data points and plotted the time history on the screen. The program then performed a Fast Fourier Transform (FFT) on the acquired data and plotted it on a separate graph. Control of the program was a collection of screen-operated pull-down menus, push buttons, switches, etc.

NATIONAL INSTRUMENTS LABWINDOWS, VERSION 2.2.1

This tool exists only for the PC platform. It is a good example of a data acquisition and analysis package that bridges the gap between text-based and visual programming. Although it is often confused with National Instruments LabVIEW, there are vast differences between them. First, LabWindows does not work under the Windows environment. It was developed before Windows even existed. LabVIEW on the other hand, only runs under Windows. Got it? Actually, National Instruments says the next version of LabWindows will run under the Windows environment.

The creation of an application in LabWindows requires the following steps:

1. Create or acquire the instrument drivers that will allow the developed application code to communicate via GPIB with the instrument(s).
2. Using the 'User Interface' function, create the 'panels' and 'menubars' that will become the visual user interface in the executing application.
3. Create the code (in C or BASIC) that will respond to user 'events' (the result of a mouse click or a keyboard entry while the application is executing), update displays, process data, and communicate (via the drivers) to the instruments.

instrument Drivers

National Instruments provides a large library of drivers for the most-popular GPIB and RS-232 controllable instruments. If a driver is to be written from scratch, a significant portion of the application development effort could be devoted to writing a driver. The first problem (assuming one knows C or BASIC) is gaining an understanding of the instrument. Control of the instrument with both its front panel and GPIB commands must be understood. The next step is to learn the LabWindows procedures that allow the creation of a driver. This part of the LabWindows package is more complex than developing screens and event handling. Driver development can be facilitated by the menu-assisted code generation capabilities of LabWindows. General instrument control functions can be selected from menus and modified for the specific instrument. In this way, the developer is isolated

from concerns about the correct syntax of the function calls. Once the driver is written or available to the programmer, developing the application is straightforward.

User Interface Function

in LabWindows, panels are the screens that contain displays, menubars, or controls (such as push buttons and slide controls). Figure 2 shows examples of the controls that can be placed on a panel. There are numerous color choices for panels and menubars. Their size is also easily modified. The panels may be nested in the sense that the menubar (or a control that is clicked) on the panel will request a subsequent panel. The secret to programming the application is to develop a naming scheme for the panels and controls that allows easy reference from code developed later.

Programming the Application

Programming an application is simplified by menus that explain and insert any C/BASIC command that is allowed by LabWindows. This means that programming experience is required, but not an in depth understanding of the syntax and semantics of the language being used. Function panels ask for variable names and build an instruction with the variable in its proper syntactical position. There are ample error messages and debugging features that help the programmer later if problems are encountered. A typical LabWindows program would have the following modules:

1. Declaring and initializing variables, library references, and initializing instruments
2. Displaying of the main panel and menubar (other panels could be called from this one)
3. Checking for and processing panel and menubar events
4. Communicating with the instruments
5. Storing, processing and displaying data

The Sample Program

The LabWindows version of the sample program consists of three panels: an Opening Screen that introduces the program and asks for an operator keystroke, a Main Panel that is used to control the test and output the results (Figure 3), and an initialization Panel that is used to initialize the instruments (Figure 4).

The Initialization Panel

The LabWindows application first displays a title screen and then the main panel shown in Figure 3. A menubar selection will display the initialization panel that is used for setting up communications with the two instruments.

The Main Panel

The main panel provides status information and allows operator input. Some of the controls indicate status, others are for operator interaction.

Status Information

The status information controls are outputs from the executing program that explain what is happening. They cannot be 'clicked on' or modified by the operator:

1. Date
2. Time-of-day
3. Function generator status (this status is updated by the initialization panel)
4. Analog to digital converter status (this status also is updated by the initialization panel)
5. Graphs of the sampled data and calculated FFT
6. Program status ('sampling' or 'stopped')
7. File name for stored data

Operator Input

These are the controls that the operator may 'click on' and modify:

1. Function generator control values (wave type, amplitude, and frequency)
2. A/D converter sampling rate
3. A button for requesting a new sample
4. A button for calculating and displaying an FFT
5. A button for sending new parameters to the function generator
6. A button for sending a new sample rate to the A/D converter

The operator selects waveform type, amplitude, and frequency for the signal generator and sampling rate for the A/D converter. The input controls were linked to logic that limited the range of acceptable values. For example, if the operator entered an amplitude of 11 volts, it would automatically be changed to 10 by the program before being sent to the instrument. Arbitrary limits for the values were made to simplify both the programming and the comparison of each computer platform and toolset.

To initiate communication with the instruments, which is a prerequisite to acquiring data, the initialization pane] is selected from the menubar. The initialization panel is shown in Figure 4. Error messages describing the problem are displayed when an instrument cannot be initialized. Control parameters can only be modified from the main panel.

Documentation

LabWindows comes with a complete set of documents explaining its use and operation. There is, however, no index, which greatly limits the value of the documentation. The on-screen help is good enough, though, that the other documentation is seldom needed.

Support

The greatest challenge in getting support when a problem occurs is getting through the telephone queue and finding a consultant who is not busy. When all the consultants are busy, a call for assistance is not always expeditiously returned. It is nearly always returned, but this may take hours.

Summary

LabWindows is an appropriate development tool when the application is required to be a stand alone program that runs on a PC. It is especially appropriate when the data acquisition is at a high rate and timing is critical. The program developer must have a rudimentary knowledge of either Basic or C. It should be pointed out that there are many non-instrumentation applications for Lab Windows. Anyone developing an application that requires a complicated set of GUIs, including simulation or process control could benefit from the use of Lab Windows.

One shortcoming of the current version of LabWindows is the non-support of the standard C libraries. This deficiency causes the programmer to spend needless time inventing a work-around for a normally available function. Reportedly, a future version of LabWindows will support the common C libraries.

VISUAL BASIC & WAVETEST

Recognizing the advantages of using GUIs for data acquisition on the PC, Wavetek Corporation created a toolset called WaveTest VIP (Visual instrument Programmer) that is used to bring instrument control into Windows applications that support Windows Dynamic Data Exchange (DDE) or Dynamic Link Library (DLL). VIP is essentially a set of functions that can be called from applications such as Microsoft Excel and Superbase IV and Windows programming languages such as Microsoft Visual BASIC, C, C++ and Quick C, Borland C, C++ and Turbo Pascal, and HP Basic for Windows.

Not only does WaveTest VIP make its instrument control, data acquisition, data analysis, and data presentation capabilities available to *any* Windows application supporting DDE or DLL, the VIP functions are completely independent of the application calling them. This gives the programmer the option of choosing an appropriate Windows application to work in rather than being locked into one particular option from beginning to end. Visual BASIC was chosen as the application to use with VIP for the sample program.

Another advantage of the VIP tool is that its modules for instrument control can be used to communicate with the same instrument over different bus interfaces. In other words, a GPIB instrument can be replaced with a VXI instrument without altering the instrument setups. The only change that needs to be made is to tell VIP to look for the instrument on the VXI bus instead of the GPIB bus. Bus interfaces supported by WaveTest VIP include GPIB, VXI, VME, and RS-232.

The Sample Program

Creation of an application using Visual Basic and WaveTest VIP requires the following steps:

1. Acquire or create (using the VIP Library Generator) instrument drivers.
2. Create instrument setups using the VIP Instrument Manager. (Instrument setups are VIP modules that will be called from Visual Basic for instrument communication.)

3. Create the necessary graphical user interfaces using the Visual Basic Form editor or the WaveTest VIP Panel Editor.
4. Write the code in Visual Basic that will respond to user events on the panels or forms, update displays, process data, and make calls to WaveTest VII' modules.

Instrument Drivers

The VIP Library Generator is used for creation of instrument drivers called instrument Library Files (ILFs). Figure 5 shows the instrument driver for the function generator. The Library Generator is WaveTest VIP's most effective and efficient component, instrument Library Files are simple to create since VIP takes care of all the lower level bus-specific interface details. This gives the Library Generator two very strong features. First, the programmer's work is reduced to filling in dialog boxes with higher level, instrument-specific commands. Second, ILFs created with VIP's Library Generator are not bus-specific so that separate drivers do not have to be written to communicate with the same instrument over different buses. Over 275 complete instrument Library Files are included in the WaveTest VIP package, so the programmer may never even have the need to use the Library Generator. Incidentally, the drivers supplied with VII' may be modified with the Library Generator to accommodate specific needs.

The Library Generator was not used in creating the sample program because both instrument library files needed were part of the WaveTest VII' instrument Driver Library. Not only were they included, they worked !

instrument Setups

The Instrument Manager has two main purposes. One is to create instrument setups. An instrument set up is a module that executes a subset of ILF commands when called from an application. The file containing the necessary setups for an application is called an instrument Command Definition (ICD) file. Two steps must be taken for the calling program to gain access to instrument setups and other VIP functions. The instrument Manager must be running in the background during program execution and the calling program must include a function call that loads the correct ICD file before making calls to these modules.

The second purpose of the instrument Manager is for testing instrument library Files. Bus trace, simulation, and interactive instrument control enable thorough testing of ILFs and instrument setup modules before they are used by the calling program. Another useful aspect of the instrument Manager is that its debugging utilities are available while the calling program is running. Bus activity, therefore, can be monitored during debugging of the calling Windows application. Figure 6 shows the Instrument Manager with the sample program ICD loaded.

Instrument setups for the sample application were created and tested in about one hour. First, the two instrument drivers from the VIP Instrument Library were loaded into the instrument Manager. The correct GPIB addresses were entered and instrument setups were created to serve the functions required by the sample program. Development time for instrument setups depends on how complex the setups are, but the process is straightforward.

Creating the User Interface

Both WaveTest VIP and Visual Basic provide tools for creating a graphical user interface. Though the Visual Basic GUI capability is more extensive and colorful than the WaveTest VIP panel, the VIP panels were used for the sample program to examine the feasibility of using VIP with a Windows application lacking its own GUI capability. Figure 7 shows a WaveTest VIP Panel. Four panels were created: a main panel, an instrument initialization panel, and two panels for the graphs of sampled data and FFT results. Development of the four panels was completed in about an hour.

The VIP panels and controls were very simple to display and interact with using Visual Basic. Though the controls available in the VII' Panel Editor are not as numerous nor colorful as the ones in Visual Basic, they were adequate for the data acquisition and instrument control purposes required by the sample program. One bug was found when using the Panel Editor. When Visual Basic is open in the background and the user chooses to exit the Panel Editor after saving a panel but without closing it, an error is incurred which requires a system reboot to resolve.

Programming in Visual Basic

The final steps in the development of the sample program involved writing the Visual Basic code to load and display the appropriate panels and to look for and process operator inputs. Essentially, each panel required a loop that looked for state changes of the push buttons on the panel.

When a state change was detected, the corresponding function was performed. Exit from the loop was controlled by a specified push button on each pane]. Fully commented code was written in about three days. The code included checks to prevent the operator from performing operations in an invalid sequence, i.e., performing an FFT before gathering data.

Documentation

Wavetek provides a Getting Started and Quick Reference Manual that contains an easy-to-follow introduction to the WaveTest VII' package. An example application is reviewed to give the user an overview of the basic components of VIP and to demonstrate its capabilities. Then, step-by-step instructions are given that allow the user to create some of these components from scratch and test them interactively. In this way, the demonstration of the application's functionality is supported by a sense of what effort goes into achieving that level of functionality. After working through examples in the Getting Started manual, a user will have a clear understanding of the purpose and basic characteristics of WaveTest VIP. Wavetek also provides two volumes of reference manuals describing the different components of VIP. The Programmer's Reference was particularly useful for looking up the commands needed to access VIP functions from the Visual Basic calling program.

Support

User support from Wavetek was excellent. If an answer to a question was not immediately forthcoming, it was quickly found. Calls were usually returned within 15 minutes. The explanations were clear and concise and the consultants were willing to spend whatever time it took for the explanation to be thoroughly understood. Except for the bug in the Panel Editor mentioned above, WaveTest VIP performed exactly as advertised and provided easy access to instrument control, data acquisition, data analysis, and data display from the Visual Basic program.

Summary

The WaveTest VII' package brings powerful instrument control capabilities to the Windows environment. The instrument Library Generator removes the headache of lower level bus interface commands from instrument driver development. The instrument Manager facilitates functional instrument control that isolates the programmer from the details of controlling the instrument. WaveTest VIP'S versatile instrument control modules can be used by any Windows DLL or DDE applications. VIP supports GPIB cards and embedded VXI controllers from several manufacturers. In addition, the next release promises to provide the same basic VIP driver development environment for AT-bus cards and instruments, overall, the WaveTest VIP toolset is simple to use and compatible with the more popular Windows applications and instrument bus interfaces. These features make it a strong candidate for instrument control. It does, however, work best when used with other software packages, e.g., Visual BASIC, EXCEL, etc., which will drive up the cost of the 'system.'

Other I'C platform Software tools

These are only two of the multitude of software tools currently available for data acquisition and processing. New programs are introduced each year. Due to the MTC's requirement for general purpose tools that interact with a wide variety of instruments and interfaces, software packages with limited functionality and/or compatibility just aren't useful.

Two packages worth noting are Laboratory Technologies' LabTech Notebook and Geotest's ATEasy. LabTech Notebook is a DOS program that uses a spreadsheet or tabular type of configuration and set-up section as well as an optional icon section. The list of drivers for internal data acquisition and GPIB cards is very impressive and includes nearly all manufacturers. It merits consideration, now that it no longer requires a hardware key.

Geotest's ATEasy is an automated test equipment program that runs under Windows. It is a menu driven 'design-your-own-language' program that seems easy to learn and use. It can incorporate the users' existing Assembly, C, or Pascal code and integrate it into its own program. It costs \$3K for the GPIB version and \$4K for the VXI version, and it does require a hardware key.

CONCLUSIONS TO PART 1

The selection of an appropriate tool depends to a large extent on the application and the customer (assuming the end user is not the program developer) who will use the product. If speed or timing is critical, LabWindows is the logical choice. LabWindows is also the logical choice if the customer wants a stand-alone executable program that requires no other software.

If the customer needs to acquire data directly into a Windows spreadsheet, WaveTest VIP is the easiest to use. This is also the best tool to use if the programmer needs the versatility of using Visual C, Visual BASIC, or any new, yet to be developed product that runs under windows.

If it is important to transport the application from one platform to another, then the only choice is one of the visual programming packages (LabVIEW for Windows or, soon, VEE for Windows).

PART 2

Part 2 will discuss the visual programming approach and discuss development of the sample program using National Instruments' LabVIEW and Hewlett Packard's Visual Engineering Environment, including VEE for Windows. Stay tuned. It's worth it.

REFERENCES

1. *An Adaptive Structure Data acquisition System using a Visual-Based Programming Language*, E. C. Baroth, D. J. Clark and R. W. Losey, Fourth AA/Air Force/NASA/OA1 Symposium on Multidisciplinary Analysis and Optimization, Cleveland, Ohio, September 21-23, 1992.
2. *Acquisition, Analysis, Control, and Visualization of Data Using Personal Computers and a Visual-Based Programming Language*, E. C. Baroth, D. J. Clark and R. W. Losey, Conference of American Society of Engineering Educators (ASEE), Toledo, Ohio, June 21-25, 1992.
3. *Diagram Compilers Turn Pictures into Programs*, Charles H. Small, EDN Special Report, June 1991, pp. 13-20.
4. *Software Makes Its Home in the Lab*, Michael Puttre', Mechanical Engineering Magazine, October, 1992, pp. 75-78.
5. *Today's Equipment Tests Tomorrow's Designs*, Debra Bulkeley, Design News Magazine, May 17, 1993, pp. 82-86.

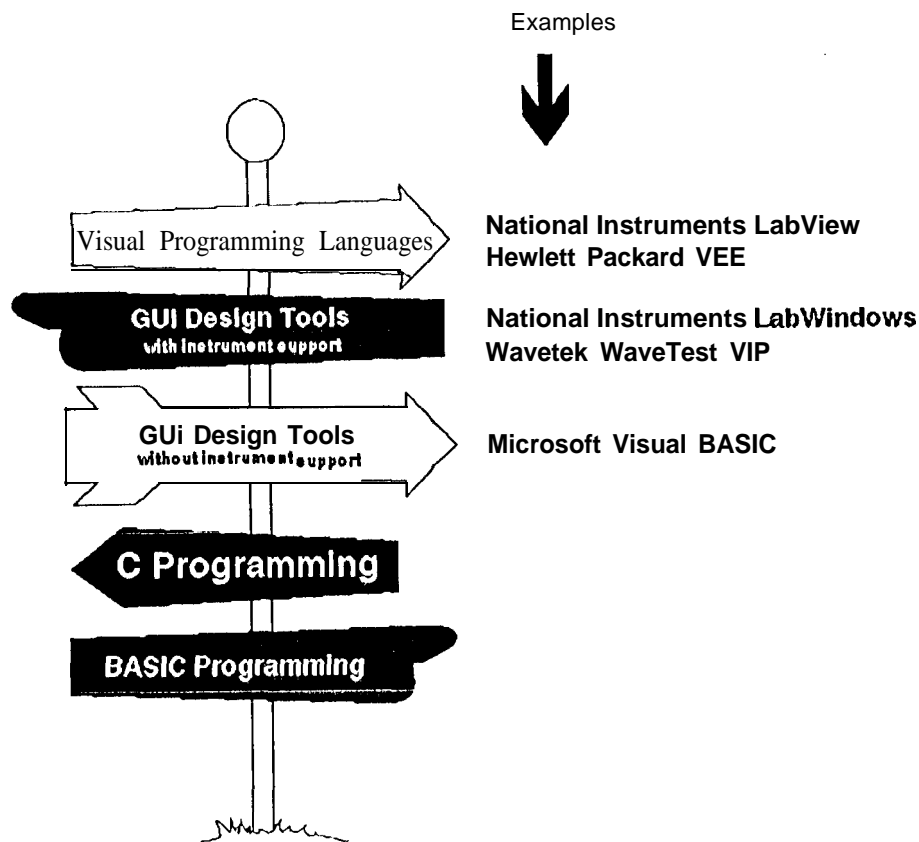


Figure 1. Instrumentation Software Crossroads

BACK TO PREVIOUS PAGE 1 ! TEXT TITLES!

Examples of Controls

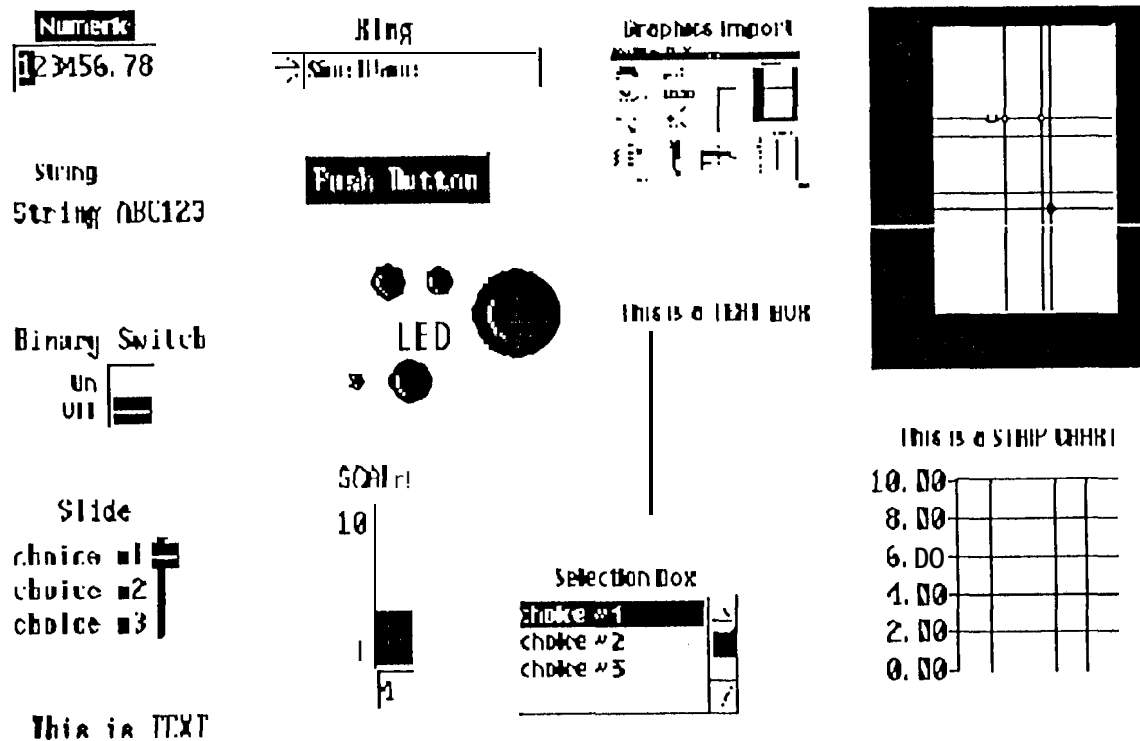


Figure 2. LabWindows PanelSample Controls

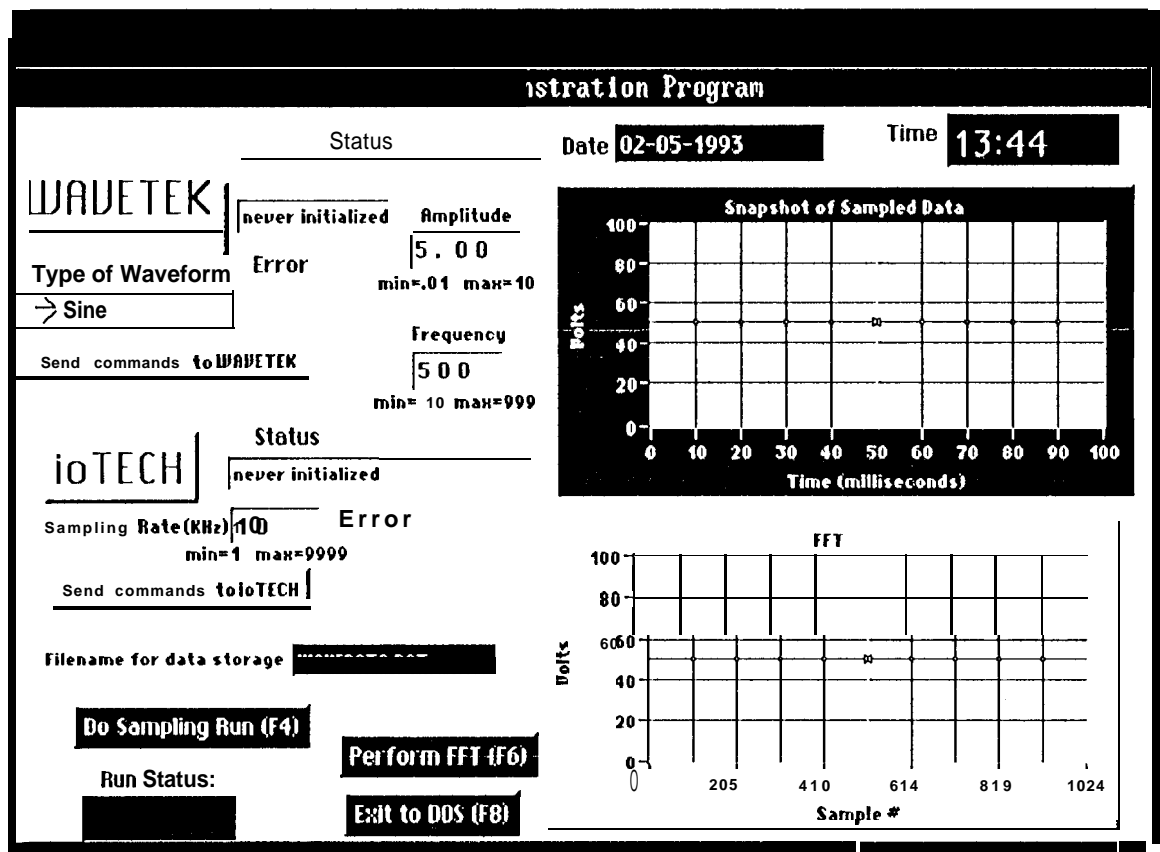


Figure 3. LabWindows Sample Program Main Panel

EXITtoMAINpanel! EXITtoDOS!

Initialize Instruments	
Wavetek Function Generator <div>Initialize</div> <div>Status</div> <div>never initialized</div> <div>Waveform Type:</div> <div>sine</div> <div>Amplitude 5.00 min=.01 max= 10</div> <div>Frequency 500 min= 10 max= 999</div>	ioTECH ADC488/16A <div>Initialize</div> <div>Status</div> <div>newer initialized</div> <div>scan count init error</div> <div>scan interval init error</div> <div>Sample Rate:</div> <div>10 KHz</div>

Figure 4. LabWindows Sample Program Initialization Panel

WaveTest Library Generator			
File Help			
WVTK23.ILF			
Edit SoftPanel Cards Special Help			
Manufacturer:	Model:	Name:	Bus Addr:
Wavetek	1 23	Synth Func Gen	" 9 'ecu
IMMEDIATE	PARAMETERS	TRIGGERING	
Execute DISCRETE	Frequency CONTINUOUS	Slope DISCRETE	Talk Mode ? QUERY
Reset DISCRETE	Amplitude CONTINUOUS	Rising Edge DISCRETE	SRQ Masks DISCRETE
MAIN		Falling Edge DISCRETE	
Function DISCRETE	Null Output DISCRETE		
Mode DISCRETE	Offset CONTINUOUS		

Figure 5. Example of WaveTest VIP instrument Driver

WaveTest VIP - mtcdemo.lcd									
File Edit Instruments Bus Help									
A to D Converter for		Synth Func Gen							

Figure 6. WaveTest VIP Instrument Manager

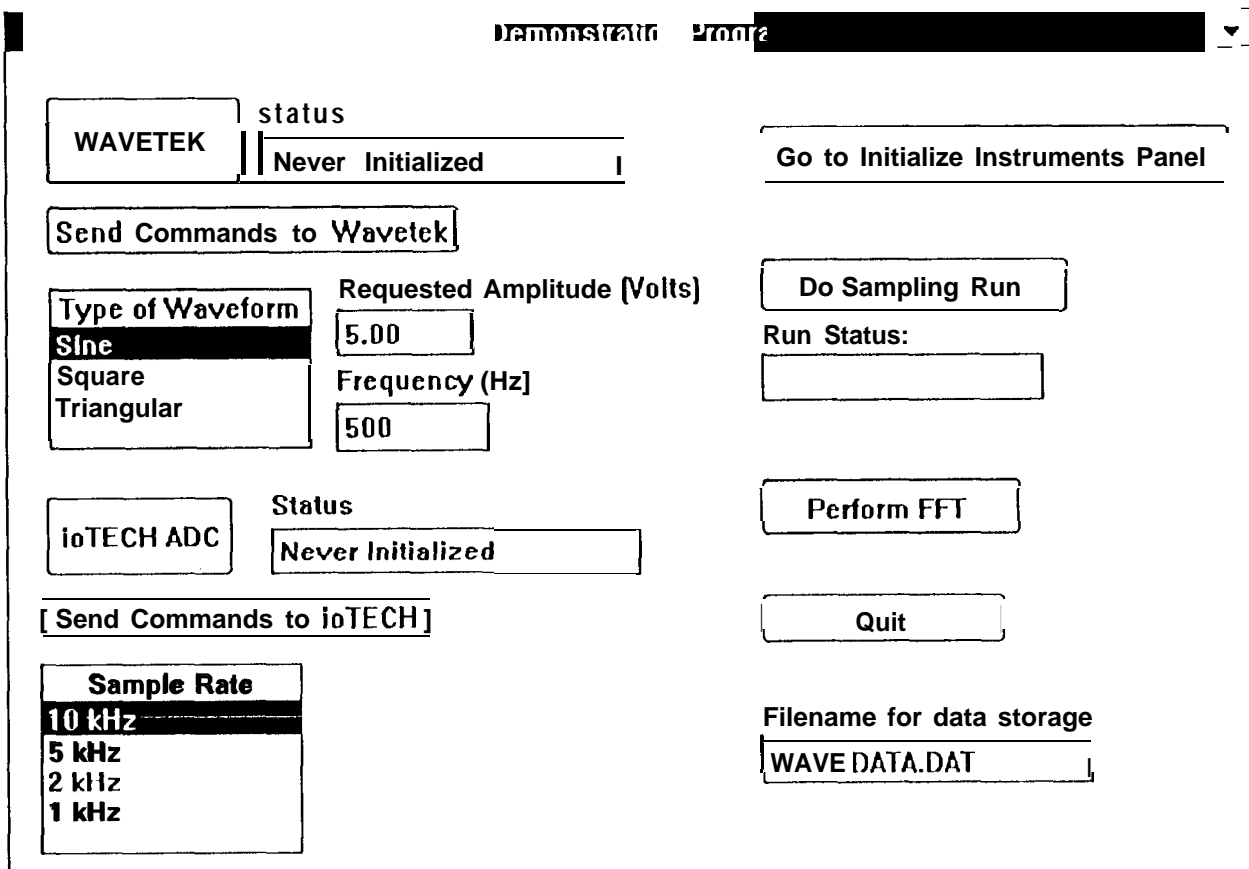


Figure 7. WaveTest VIP Panel

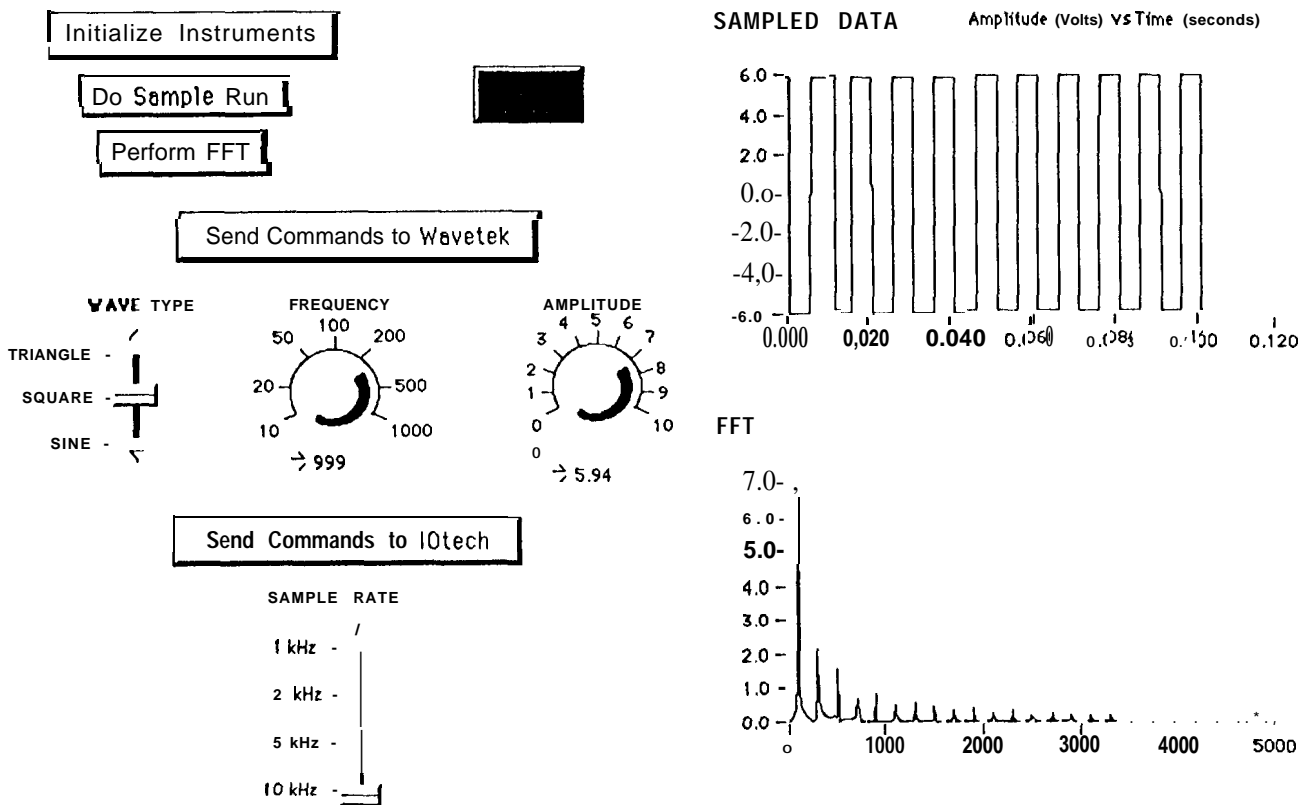


Figure 8. LabVIEW Sample Program Front Panel

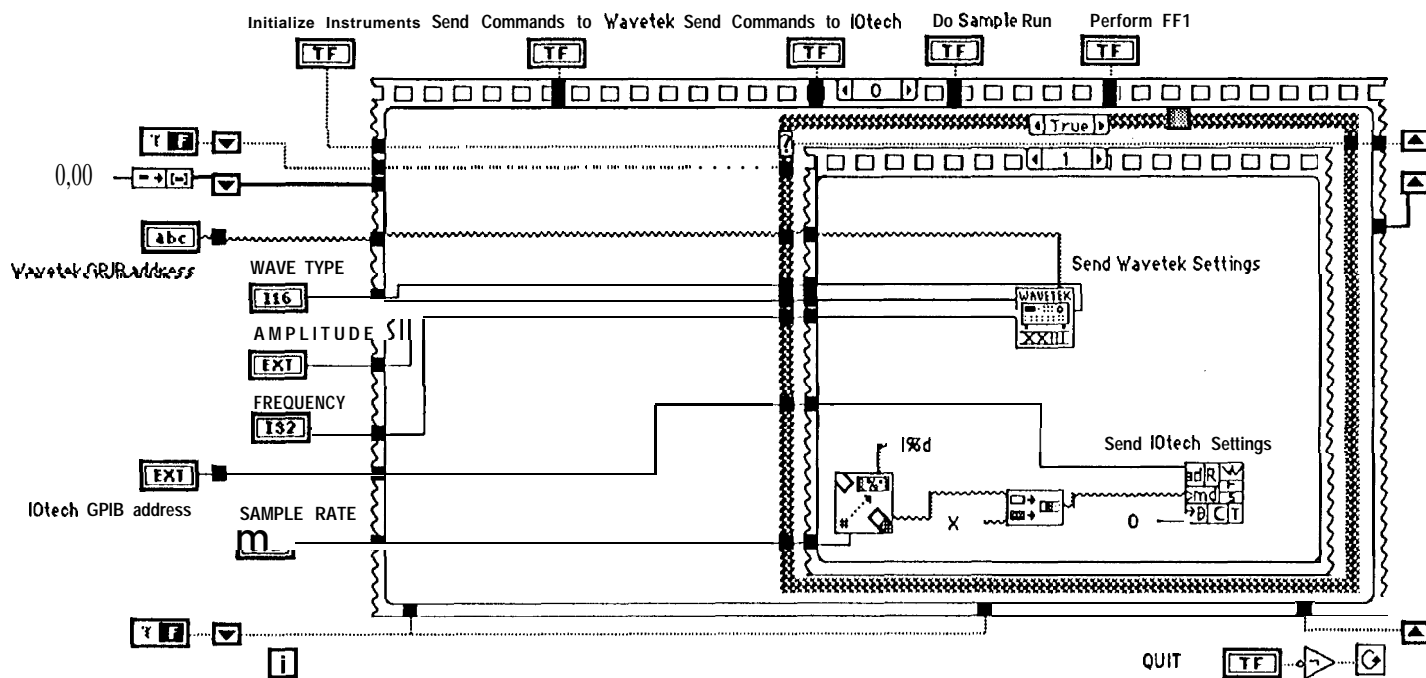


Figure 9. Sample Program LabVIEW Diagram

WAVETEK 23 SYNTHESIZED FUNCTION GENERATOR

5
GPIB ADDRESS

→ 5.00
AMPLITUDE

→ 1.000E4
FREQUENCY

→ 0.00
OFFSET

FUNCTION	MODE	TRIGGER SLOPE
D C - /	SYNTHESIZED - /	NEGATIVE - /
RAMPDN -	CLOCK -	POSITIVE -
RAMPUP -	GATEDHAVER -	
TRIANGLE -	TRIG HAVER -	
SQUARE -	GATED -	
SINE -	TRIGGERED -	
	CONTINUOUS -	

Figure 10a. LabVIEW Sample Program Instrument Driver Panel

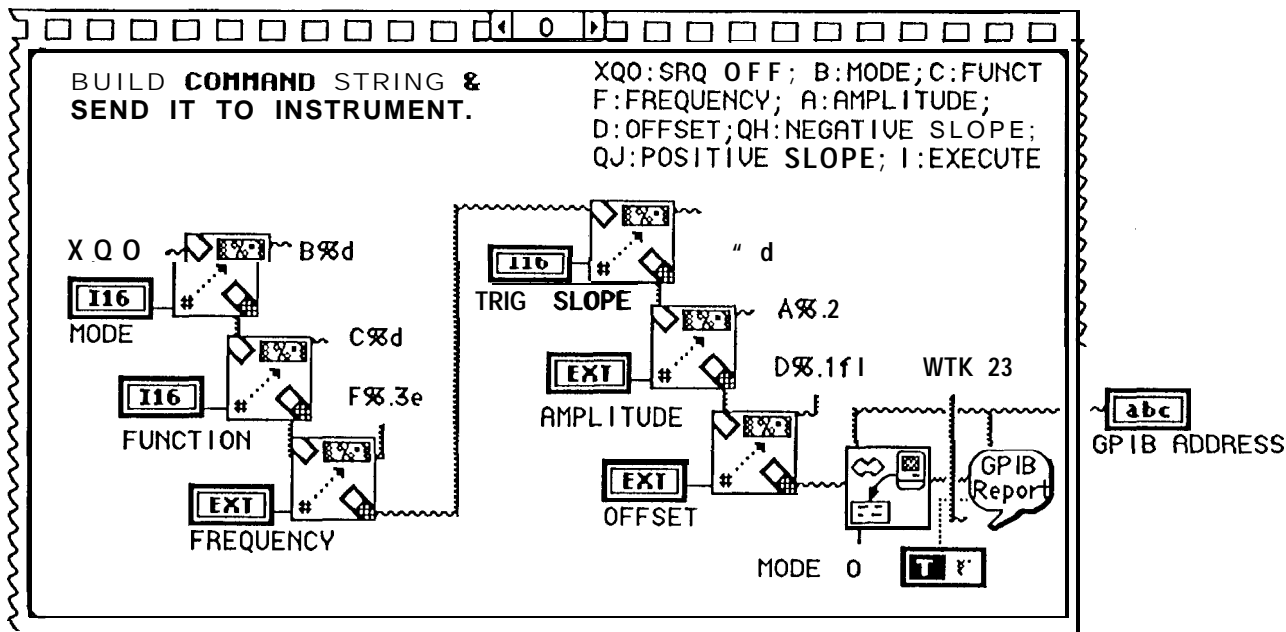


Figure 10b. LabVIEW Sample Program Instrument Driver Diagram

MTC Demonstration Program

WAVETEK
 Status: never initialized
 Error: 0
 Requested Amplitude: 5.00 (min=.01 max=10)
 Frequency: 500 (min=10 max=999)
 Send Commands To WAVETEK

ioTECH
 Status: never initialized
 Sample Rate: 10 kHz
 Send Commands To ioTECH

Filename for storage:
 DoSamplingRun (F4)
 Run Status:
 Perform FFT (F6)
 Exit to DOS (F8)

Snapshot of Sampled Data
 A graph showing voltage (0.0 to 1.0) versus Time (0 to 8 milliseconds).

FFT
 A graph showing the Fast Fourier Transform (Sample # 0 to 4).

Figure 7a. Visual BASIC Form (EXTRA)

4.4.9

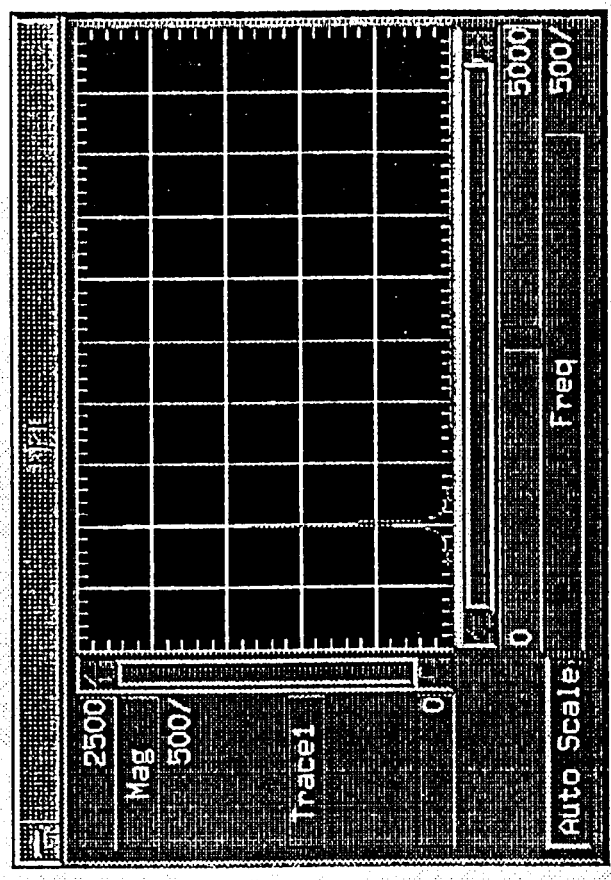
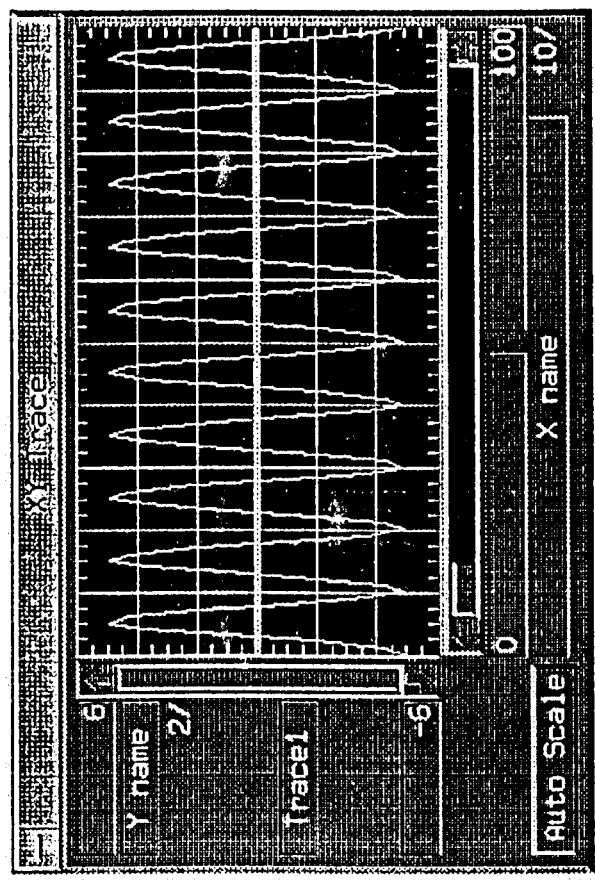
Initialize Instruments

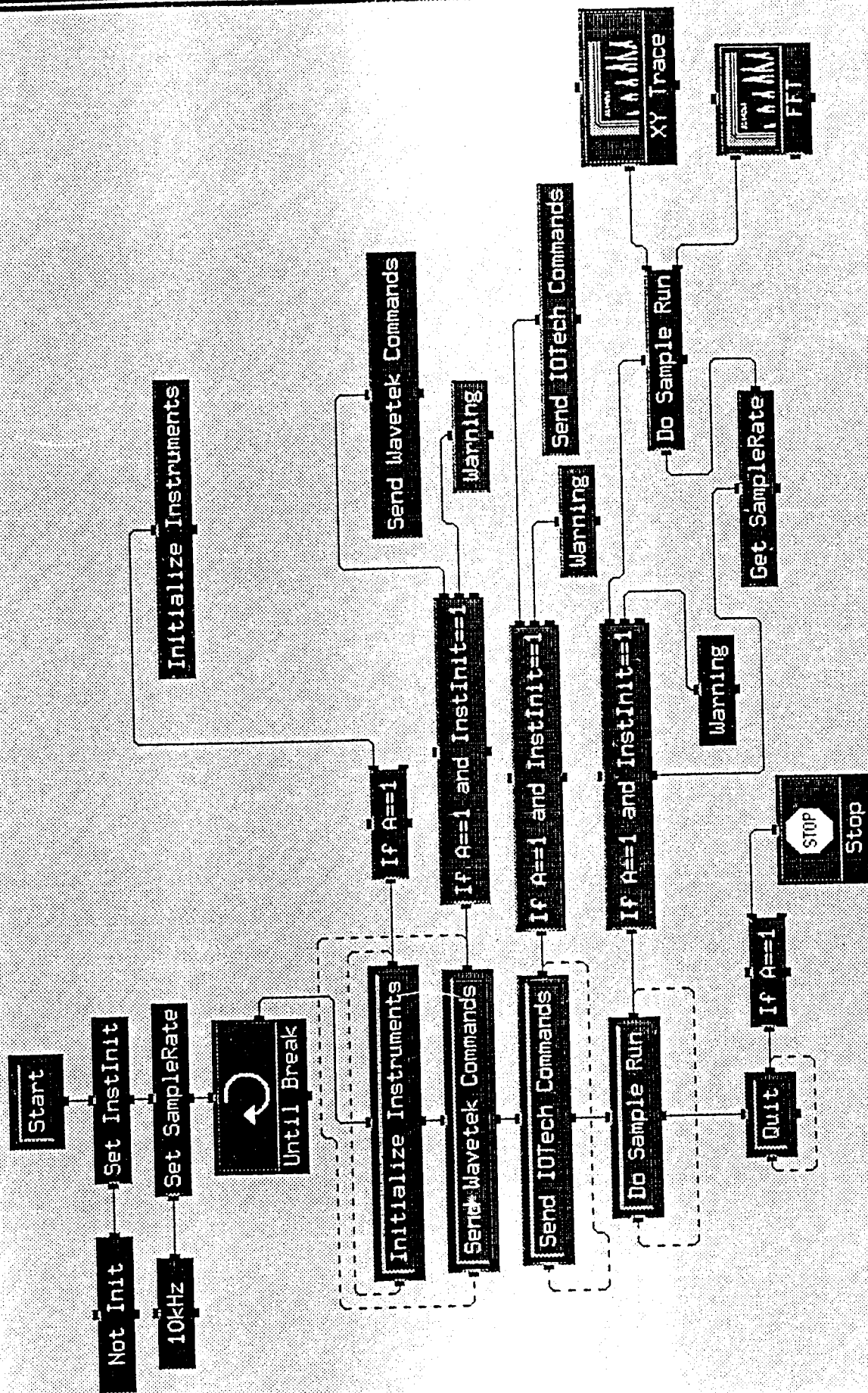
Send Wavetek Commands

Send IOTech Commands

Do Sample Run

Quit





Initialize Instruments

Send Wavetek Commands

Send IOTec


Do Sample

Function

- ☒ sine
- ☐ square
- ☐ triangle

Frequency

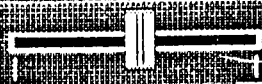
400 999



10

Amplitude


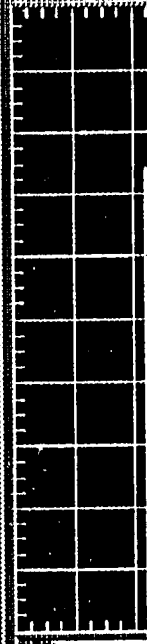
5.00E+00 10.2



100m

XY Trace

Y name 27



Auto Scale Freq

0 1000 100

OK